บุญเลิศ เอี่ยมทัศนา

เมตาไฟล์
: ไฟล์แห่งอนาคต

แนะนำตัวเมตาไฟล์
ไฟล์เก็บคำสั่ง
ที่ทำให้เกิดภาพ.....

ในกระบวนการของวินโดวส์ไม่ว่าจะเป็นหน้าต่างเมนู ไดอะล็อกบ็อกซ์ บิตแม็พ เรื่อยไปถึง DDE และ OLE ในคู่มือการเขียนโปรแกรมของวินโดวส์จะนำมากล่าวถึงอย่างค่อนข้างละเอียดพอสมควร พรั่งพร้อมด้วยตัวอย่าง ใช้เป็นคู่มือในการเขียนโปรแกรมได้

แต่สำหรับเมตาไฟล์ (Metafile) กล่าวถึงไว้น้อยมากคือ เป็นหัวข้อ ๆ หนึ่งเท่านั้น ส่วนของบิตแม็พให้ถึง 1 บทเต็ม ทั้งที่ทั้งคู่ต่างก็มีความสำคัญไม่ยิ่งหย่อนกว่ากัน เพราะใช้สำหรับเก็บภาพทั้งคู่ (สำหรับผมเมตาไฟล์สำคัญกว่า) และเมตาไฟล์มีเรื่องที่จะกล่าวถึงมากกว่าจึงอ้อมแอ้มกล่าวไว้ไม่หมด จนดูเหมือนเป็นของส่วนตัวของไมโครซอฟต์

เท่าที่หามาได้

และที่น่าประหลาดคือ หนังสือภาษาอังกฤษที่กล่าวถึงการเขียนโปรแกรมบนวินโดวส์ ไม่ว่าในภาษาซีหรือปาสคาล 8 ใน 10 เล่ม (ใช้เวลาร่วม 3 ชั่วโมงกับร้านหนังสือใหญ่ 2 ร้าน) ไม่มีคำว่าเมตาไฟล์ คือ Metafile ที่กล่าวถึงก็เป็นเพียงหัวข้ออีกเช่นกัน และไม่มีเล่มใดเลยที่กล่าวถึง Placeable Metafile ซึ่งเป็นมาตรฐานของเมตาไฟล์มาตรฐานหนึ่ง

บางท่านอาจสงสัยว่าแล้วผมนำมาจากไหน ผมได้มาจาก Help ในเทอร์โบปาสคาลและการจับโน่นผสมนี่ทดลองเขียนโปรแกรมดู ได้บ้างไม่ได้บ้างเรื่อยมา ใช้เวลานานพอสมควร ซึ่งบางเรื่องหากมีคำอธิบายเพียง 2 ประโยคผมจะได้ไม่ต้องใช้เวลานานเป็นสัปดาห์

แต่ก็ไม่ได้มาทั้งหมดครับ ขาดบางเรื่อง ในส่วนของเพลซเอเบิลเมตาไฟล์ซึ่งดูเหมือนจะเป็นยิ่งกว่าของส่วนตัวของไมโครซอฟต์ (แปลกครับเป็นผู้กำหนดมาตรฐานแต่ไม่แจ้งให้ทราบ) จึงขอเรียนให้ทราบไว้ก่อน

แนะนำตัว

เพราะมีการกล่าวถึงกันน้อยผมจึงขอแนะนำตัวก่อน เมตาไฟล์คือ ไฟล์ที่เก็บคำสั่งครับ เมื่อเทียบกับบิตแม็พคือ

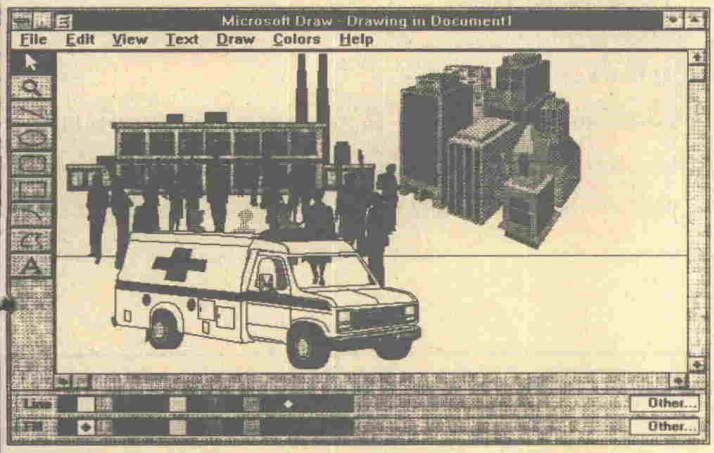
– บิตแม็พเก็บภาพอันเกิดจากคำสั่งภาพ

– เมตาไฟล์เก็บคำสั่งที่ทำให้เกิดภาพ

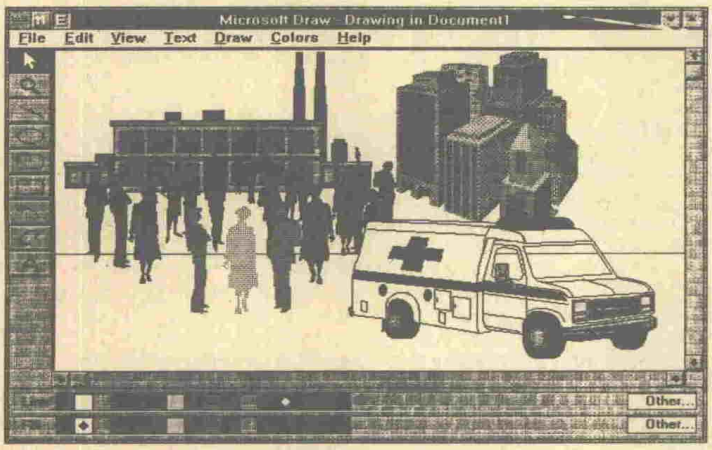
คำสั่งที่ทำให้เกิดภาพคือฟังก์ชันของ GDI ของวินโดวส์ ดังนั้นไฟล์ของบิตแม็พคือไฟล์ที่ตายแล้ว ส่วนของเมตาไฟล์เรานำมาเปลี่ยนแปลงแก้ไขได้ และมีเรื่องอื่นอีกซึ่งเมตาไฟล์ทำได้แต่บิตแม็พเช่นใน Paintbrush ทำไม่ได้ (ทำได้ครับแต่ต้องเขียนโปรแกรมเป็นพิเศษ ซับซ้อนกว่า ยาวกว่า) คือ

ย้ายภาพย่อย

ดังเช่นในรูปที่ 1 เป็นภาพอันเกิดจากการแสดงภาพในเมตาไฟล์ (หรือจะเกิดจากการวาดได้เช่นกัน เพียงแต่ผมไม่ได้วาดภาพนี้เอง) ของโปรแกรม Microsoft Draw ซึ่งก็ดูเหมือนภาพบิตแม็พแต่ไม่เหมือนกันตรงที่ถ้าเป็นบิตแม็พภาพทั้งหมดนี้คือภาพเดียวกัน ส่วนเมตาไฟล์จะประกอบด้วย 5 ภาพย่อยคือ



รูปที่ 1
ภาพจากการแสดงภาพในเมตาไฟล์



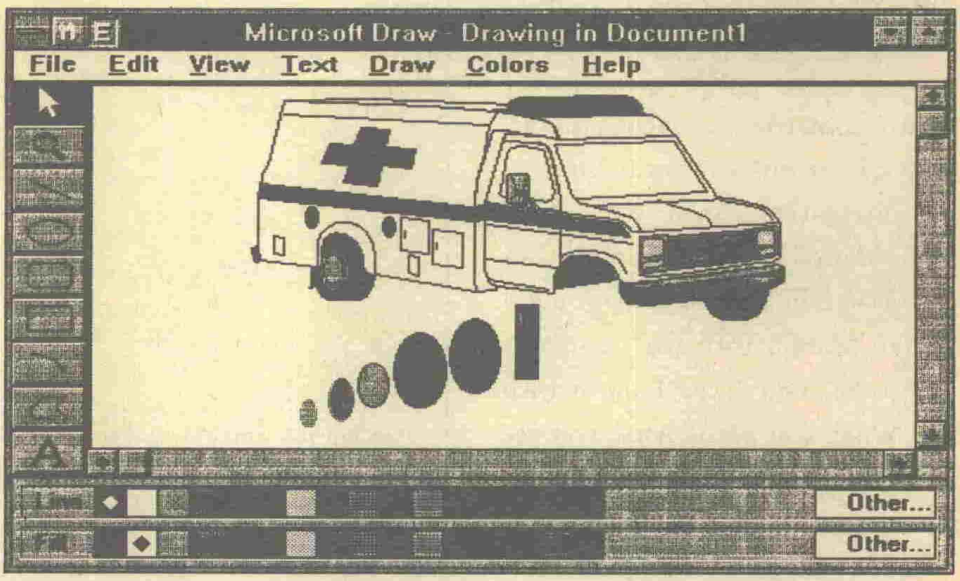
รูปที่ 2
แสดงการเลื่อนภาพ

โรงงาน (ทางซ้าย) สำนักงาน (ทางขวา) ฝูงชน รถยนต์ และที่ผมลากเป็นเส้นตรงเส้นเดียว

ภาพย่อยทั้ง 5 ภาพนี้ต่างเป็นอิสระต่อกัน ดังที่ผมย้ายรถยนต์ไปทางขวาในรูปที่ 2 ซึ่งผมเพียงแต่คลิ๊กเม้าส์ที่รถยนต์แล้วเลื่อนไปไว้ ณ ตำแหน่งใหม่เท่านั้น

แยกส่วนประกอบของภาพ

รูปที่ 3 รถยนต์คันเดิมครับ ผมนำมาถอดล้อออก โดยคลิ๊กเม้าส์เลื่อนชิ้นส่วนออกมาทีละชิ้นเช่นกัน ถึงแม้ภาพนี้จะไม่ได้เขียนไว้เพื่อแสดงการถอดประกอบ ก็คงพอที่ให้เห็นแนวทาง



รูปที่ 3 แสดงการเลื่อนส่วนประกอบของภาพ

ชิ้นส่วนอื่น ๆ ก็สามารถถอดออกได้เช่นกัน และขอเรียกชิ้นแต่ละชิ้นว่าแพตเทอร์น คือ เราสามารถรวมแพตเทอร์นเป็นภาพย่อยได้ เช่น เป็นภาพล้อ 1 ล้อ ถอดออกได้ทั้งล้อ

แสดงภาพต่อเนื่อง

ข้อแตกต่างที่สำคัญอีกประการหนึ่งคือ เราสามารถแสดงภาพในเมตาไฟล์ที่อยู่ในไฟล์เดียวกันทั้งหมดหรือไม่ทั้งหมดก็ได้ เช่น อาจวาดภาพแผนที่ประเทศไทยมีขอบเขตของประเทศ ภูเขา และแม่น้ำเก็บไว้ ในการแสดงอาจให้แสดงเฉพาะขอบเขต หรือต่อไปให้แสดงภาพภูเขา และต่อไปให้แสดงแม่น้ำ ความช้าเป็นผลดีครับ เพราะจะให้ความรู้สึกที่ดีกว่าแก่ผู้ชม

การเก็บภาพ

เก็บเป็นคำสั่งซึ่งจะเรียกว่าเป็นอะเรย์ก็ไม่ถนัดจึงขอเรียกว่าเป็นแถวของคำสั่ง ซึ่ง

– เราสามารถตรวจ แก้ และถอดถอนคำสั่งให้ ทำให้ภาพเปลี่ยนไป

– ขนาดของไฟล์จะเล็กกว่าของบิตแม็พพอสมควรคือ น้อยกว่าครึ่งหนึ่ง และไม่ขึ้นกับขนาดของภาพ หากขึ้นอยู่กับความละเอียดคือ จำนวนคำสั่ง

– ในเมตาไฟล์ สามารถเก็บภาพบิตแม็พโดยเก็บคำสั่งที่นำภาพมาใช้

ความเร็ว

การแสดงภาพของเมตาไฟล์เป็นการวาดภาพใหม่จึงช้ากว่าบิตแม็พ เป็นเรื่องเดียวที่เมตาไฟล์ด้อยกว่าบิตแม็พ แต่ก็เหมาะสำหรับงานบางงานเช่น เมื่อต้องการให้แสดงภาพต่อเนื่อง และสำหรับเมื่อต้องการให้แสดงภาพกับจอภาพ หากไม่ต้องการให้เห็นการวาดภาพอาจให้แสดงบนบิตแม็พก่อนแล้วจึงลอกมาออกจอภาพ

ขนาดของภาพ

เมื่อเราใช้งานบิตแม็พ เราไม่ค่อยคำนึงถึงมาตรา (mapping made) เท่าไร แต่ของเมตาไฟล์ขนาดของภาพจะขึ้นกับมาตรามาก เช่น เมื่อเราใช้คำสั่ง Rectangle (adc, 100, 200, 300, 400) เก็บไว้ในเมตาไฟล์ ค่า 100 ถึง 400 คือ ค่าหน่วย (unit) เท่านั้น เมื่อเรานำมาแสดงหากใช้มาตราเป็น mm__text ก็จะได้ภาพที่ใหญ่กว่ามาตรา mm__LoMetric และเราสามารถเลือกมาตราซึ่งหมายถึงมาตราส่วนให้เหมาะกับอุปกรณ์แต่ละชนิดได้

จะเห็นว่าเราสามารถใช้งานเมตาไฟล์ได้กว้างกว่าบิตแม็พมาก ไม่จำกัดอยู่แค่เพียงการแสดงภาพเท่านั้น และเมื่อมาถึงยุคของ True Type Font คำพูดที่ว่า What you get is what you see จะต้องเปลี่ยนเป็น What you get is bilter than what you see ยุคนี้อาจเป็นยุคของบิตแม็พแต่อนาคตคือ ของเมตาไฟล์ครับ

ไฟล์ที่ไม่เหมือนไฟล์

เมตาไฟล์ก็คือ ไฟล์ และยิ่งกว่าไฟล์ธรรมดาอีกคือ เป็นได้ทั้งไฟล์ในดิสค์และในหน่วยความจำ แต่มีลักษณะไปทางเครื่องบันทึก (recorder) การใช้งานจึงแบ่งออกเป็น 2 ภาค คือ

– ภาคบันทึก (record)

– ภาคแสดง (play) ในที่บางแห่งจะขอใช้คำว่า เพลย์ ด้วย ตามความเหมาะสม

มีขั้นตอนการใช้งานตามที่จะกล่าวต่อไปนี้ ขอให้ดูโปรแกรมในลิสติงที่ 1 ประกอบด้วยครับ โดยผมจะให้ชื่อกิจกรรมที่อ้างถึงไว้ในวงเล็บ กล่าวคือ

ลิสติงที่ 1

```
PROGRAM   SimMetaFile;  { simmtf.pas }

{$R simmtf.res}

USES   WinTypes,WinProcs,WObjects;
CONST  appname  = 'simmtf';
       thetitle = 'Simple Meta File';
       menuname = 'simmtf_menu';

       cm_New  = 100;  cm_Make = 200;  cm_Load = 300;


TYPE
  WndProCP = ^WndProC;
  WndProC = OBJECT(TWindow)
    hPlay : THandle;
 constructor  Init(apar : PWindowsObject; atitle : PChar);
 procedure  Destroy; virtual;
 procedure  Paint(paintDC: HDC;
                  var PaintInfo: TPaintStruct); virtual;
 procedure  cmNew(var msg : TMessage);
             virtual cm_First + cm_New;
 procedure  cmMake(var msg : TMessage);
             virtual cm_First + cm_Make;
 procedure  cmLoad(var msg : TMessage);
             virtual cm_First + cm_Load;
 procedure  DrawGraph(adc,mtdc : HDC);
            END;


(* ------------------ WndProC  Method ----------------- *)

CONSTRUCTOR  WndProC.Init(apar : PWindowsObject; atitle : PChar);
BEGIN
  TWindow.Init(apar,atitle);
  attr.menu := LoadMenu(HInstance, menuname);
  hPlay := 0;
END;


PROCEDURE  WndProC.Paint(paintDC: HDC;
                         var PaintInfo: TPaintStruct);
BEGIN
  if  hPlay <> 0  then  PlayMetaFile(paintDC, hPlay);
END;


PROCEDURE  WndProC.Destroy;
BEGIN
  if  hPlay <> 0  then  DeleteMetaFile(hPlay);
  TWindow.Destroy;
END;


PROCEDURE  WndProC.cmNew(var msg : TMessage);
```

ลิสติงที่ 1 (ต่อ)

```
  BEGIN
    if  hPlay <> 0  then  DeleteMetaFile(hPlay);
    hPlay := 0;
    InvalidateRect(HWindow, nil, true);
  END;

  PROCEDURE  WndProC.cmMake(var msg : TMessage);
  VAR  recDC,adc : HDC;
  BEGIN
    if  hPlay <> 0  then  Exit;
    adc := GetDC(HWindow);
    recDC := CreateMetaFile('e:\simmtf.wmf');
    DrawGraph(adc,recdc);
    ReleaseDC(HWindow, adc);  hPlay := CloseMetaFile(recDC);
  END;

  PROCEDURE  WndProC.cmLoad(var msg : TMessage);
  BEGIN
    if  hPlay = 0  then
      begin  hPlay := GetMetaFile('e:\simmtf.wmf');
             InvalidateRect(HWindow, nil, true)  end;
  END;


  PROCEDURE  WndProC.DrawGraph(adc,mtdc : HDC);
  VAR  open,npen : HPen;  oldbkco : TColorRef;
       obrsh,nbrsh : HBrush;
  BEGIN
    npen := CreatePen(ps_Solid, 10, RGB(0,0,255));
    open := SelectObject(adc, npen);
    SelectObject(mtdc, npen);

    nbrsh := CreateSolidBrush(RGB(192,192,192));
    obrsh := SelectObject(adc, nbrsh);
    SelectObject(mtdc, nbrsh);

    RoundRect(adc, 100,50, 300,200, 40,40);
    RoundRect(mtdc, 100,50, 300,200, 40,40);


    SetBkColor(adc, oldbkco);  SetBkColor(mtdc, oldbkco);
    SelectObject(adc, obrsh);  DeleteObject(nbrsh);
    SelectObject(mtdc, obrsh);
    SelectObject(adc, open);  DeleteObject(npen);
    SelectObject(mtdc, open);
  END;


  (* ------------------  TheAppC Class  ---------------- *)

  TYPE
    TheAppC = OBJECT(TApplication)
      procedure  InitMainWindow; virtual;
               END;


  PROCEDURE  TheAppC.InitMainWindow;
  BEGIN
    MainWindow := New(WndProCP, Init(nil,thetitle));
  END;

  (* --------------------  MAIN PROGRAM  ----------------- *)

  VAR   app :  TheAppC;

  BEGIN
    app.Init(appname);
    app.Run;
    app.Done;
  END.
```

เปิดไฟล์ใหม่

ให้เปิดไฟล์ด้วย CreateMetaFile (ใน cmMake) ซึ่งจะเป็นการเปิดไฟล์ในลักษณะ Rewrite หากใช้ชื่อไฟล์เป็น nil จะเป็นการเปิดไฟล์ในหน่วยความจำ

บันทึก

เมื่อเปิดไฟล์แล้วจะให้ค่าดีซีมาซึ่งขอเรียกว่า recDC ตามชื่อตัวแปร มีสถานะเป็นทั้งไฟล์คือ เมตาไฟล์ภาค บันทึก กับ ดีซี (device context) เพื่อใช้กับฟังก์ชันทาง GDI ของวินโดวส์ เช่นเดียวกับดีซีทั้งหลาย แต่จะเป็นการบันทึกเก็บไว้ในไฟล์ ดังใน DrawGraph ซึ่งได้เขียนโปรแกรมจำลองการทำงานของโปรแกรมที่ให้ผู้ใช้วาดภาพ คือ

– จะใช้คำสั่งวาดภาพกับดีซีของหน้าต่างคือ adc กับดีซีของเมตาไฟล์คือ metDC คู่ขนานกันไป

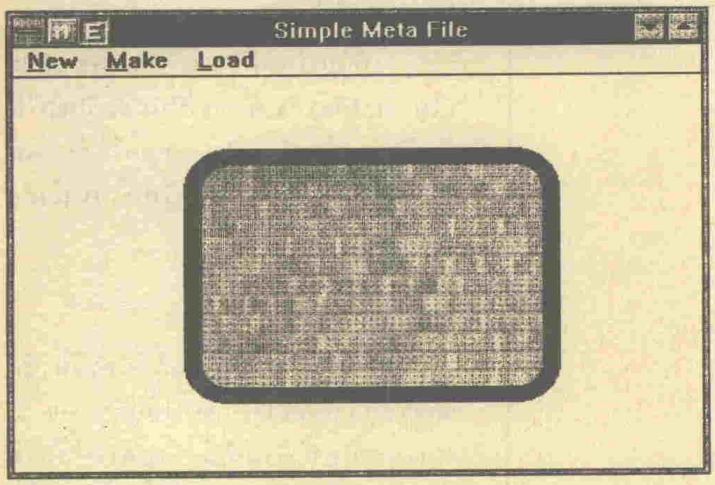
– ตามโปรแกรมนี้ได้ให้กำเนิดปากกาสีน้ำเงินและแปรงสีเทาให้แก่ดีซีทั้งสอง เพื่อใช้ในการวาดรูปสี่เหลี่ยมมุมมนกับดีซีทั้งสองเช่นกัน ทำให้เกิดภาพตามรูปที่ 4

ปิดไฟล์

เมื่อบันทึกภาพเสร็จแล้วให้ปิดไฟล์ด้วย CloseMetaFile (ใน cmMake) จะได้ค่าแฮนเดิลคืนมาเป็นเมตาไฟล์ภาคแสดง ซึ่งขอเรียกว่า hPlay ตามชื่อตัวแปร และ recDC จะหมดสถานะของการเป็นตัวแปร นั่นคือ เมื่อปิดไฟล์แล้วเราจะบันทึกต่อไม่ได้

เพลย์

เมื่อต้องการแสดงภาพในเมตาไฟล์ให้เพลย์ด้วยฟังก์ชัน PlayMetaFile กับดีซีของอุปกรณ์ที่ต้องการนำภาพไป



รูปที่ 4
ผลการรันของโปรแกรม SimMetaFile

แสดงและด้วย hPlay ซึ่งใน Paint ให้เพลย์กับดีซีของหน้าต่างคือ paintDC จะเหมือนกับนำคำสั่งที่ให้บันทึกไว้มาปฏิบัติทีละคำสั่ง (การแสดงภาพจึงช้ากว่าบิตแม็พ)

ยกเลิก

เมื่อไม่ต้องการใช้เมตาไฟล์นี้แล้ว เช่นใน Destroy เมื่อผู้ใช้ปิดหน้าต่าง หรือใน cmNew ซึ่งสมมติว่าผู้ใช้ต้องการวาดภาพเก็บในไฟล์ใหม่ ให้ยกเลิก hPlay ด้วย DeleteMetaFile ซึ่งจะเป็นการยกเลิกตัวแปรนี้ (รวมทั้งไฟล์ในหน่วยความจำ) เท่านั้น ไม่เกี่ยวกับไฟล์ในดิสก์

เปิดไฟล์เก่า

สามารถเปิดไฟล์ที่มีอยู่แล้วมาเพลย์ได้ด้วย GetMetaFile ดังใน cmLoad ได้เป็นค่าแฮนเดิลคือ hPlay

ฟังก์ชันของเมตาไฟล์

การใช้งานเมตาไฟล์โดยทั่วไปมีเท่านี้ มีฟังก์ชันที่ยังไม่ได้กล่าวถึงอีก 5 ฟังก์ชัน แต่ขอนำเรื่องที่จำเป็นต้องทราบมากล่าวถึงก่อนคือ

– ขนาดเป็นเวอร์ก การนับขนาดของหน่วยความจำโดยทั่วไปวัดเป็นไบต์ แต่ของเมตาไฟล์นับเป็นเวอร์ก เช่น 7 เวอร์ก (คือ 14 ไบต์)

– แถวของเรคอร์ด การเก็บคำสั่งวาดภาพตามที่กล่าวมา จะเก็บต่อเนื่องกันไปเป็นแถวของเรคอร์ด เรคอร์ดหนึ่ง ๆ คือคำสั่งหนึ่ง ๆ จึงมีขนาดไม่เท่ากัน และจะเก็บต่อจากส่วนหัวของเมตาไฟล์ตามที่จะกล่าวต่อไป

ฟังก์ชันของเมตาไฟล์ทั้งหมดจะดูได้จากหัวข้อ MetaFiles Functions ที่ยังไม่ได้นำมากล่าวถึงคือ

– CopyMetaFile ก๊อบปี้เมตาไฟล์ภาคแสดงคือ hPlay ให้แก่ไฟล์ตามชื่อไฟล์หรือ nil เมื่อเป็นหน่วยความจำ ได้ค่าคืนมาเป็น hPlay เช่นกัน

– EnumMetaFile เพื่อการดำเนินการกับเมตาไฟล์เป็นเรคอร์ด (จะแสดงตัวอย่าง)

– PlayMetaFileRecord แสดงเมตาไฟล์เป็นเรคอร์ดใช้ร่วมกับ EnumMetaFile

– GetMetaFileBits ทำแถวของเรคอร์ดให้เป็น hPlay (เช่นเมื่อโหลดเฉพาะส่วนนี้อย่างไฟล์ธรรมดาทำไมต้องโหลด จะกล่าวต่อไปครับ)

– SetMetaFileBits ทำ hPlay ให้เป็นแถวของเรคอร์ด

– SetMetaFileBitsBelter ทำให้ hPlay ให้เป็นแถวของเรคอร์ดเพื่อใช้ในเรื่องของ OLE (object linking and embedding)

ทำเป็นสกุล

จะเห็นว่าไม่มีฟังก์ชันใดเลยที่ให้เราเปิดไฟล์เพื่อเขียนต่อท้าย คือ เมื่อเปิดไฟล์แล้วจะให้ค่าเป็นค่าดีซี และข้อสำคัญก็คือ ในขณะที่ให้ผู้ใช้วาดภาพซึ่งเราจะต้องบันทึกลง recDC ตามไปด้วยนั้น หากต้องเพนต์จะต้องนำภาพใน recDC มาแสดง ซึ่งจะต้องปิดเมตาไฟล์ภาคบันทึกเป็นเมตาไฟล์ภาคแสดงคือ ทำเป็น hPlay ก่อน ทำให้ใช้ recDC ต่อไปไม่ได้อีก

ผมจึงคิดทำเป็นสกุลของเมตาไฟล์ให้มองเห็นเป็นไฟล์ธรรมดาที่สามารถเปิด ปิด และบันทึกต่อท้ายได้ แต่เพื่อไม่ต้องทำเป็นสกุลมากชั้นนัก โปรแกรมนี้จะมีส่วนของเอดิเตอร์ผสมอยู่ด้วย และบังเอิญผมยังไม่ได้กล่าวถึงเรื่องมาตราเลยจึงขอนำมากล่าวถึงก่อน รวมทั้งโครงสร้างไฟล์ของเมตาไฟล์ด้วย

โครงสร้างไฟล์ของเมตาไฟล์

การใช้งานเมตาไฟล์โดยทั่วไปมีเพียงเท่านี้ แต่ถ้าต้องการเขียนโปรแกรมต่อไปซึ่งจะเกี่ยวกับการใช้ฟังก์ชัน EnumMetaFile ด้วย จะต้องทราบถึงการเก็บข้อมูลในเมตาไฟล์ ซึ่งแยกออกได้เป็น 2 ส่วนคือ ส่วนหัว กับแถวของเรคอร์ด กล่าวคือ

ส่วนหัว

อยู่ต้นไฟล์มีแบบข้อมูลเป็น TMe-

```
TMetaHeader = record
  mtType          : Word
  mtHeadSize      : Word
  mtVersion       : Word
  mtSize          : LongInt
  mtNoObjects     : Word
  mtMaxRecord     : LongInt
  mtNoParameters  : Word
              end;

TMetaRecord = record
  rdSize: Longint;
  rdFunction: Word;
  rdParm: array[0..0] of Word;
              end;
```

รูปที่ 5 แบบข้อมูลเรคอร์ดที่ใช้แสดงโครงสร้างไฟล์ของเมตาไฟล์

taHeader ดังแสดงในรูปที่ 5 ประกอบด้วยฟิลด์ต่าง ๆ ดังนี้

– mtType ถ้าเป็น 0 หมายถึงเป็นไฟล์ในหน่วยความจำ ถ้าเป็นไฟล์ในดิสค์จะมีค่าเป็น 1

– mtHeadSige คือ ขนาดของตัวมันเองเป็นเวอร์ก คือมีค่าเป็น 9

– mtVersion เป็นรุ่นสำหรับวินโดวส์ 3.0 ขึ้นไปมีค่าเป็น 0300h คือ 768 รุ่นก่อนมีค่าเป็น 0100 h คือ 256

– mtSige ขนาดของไฟล์เป็นเวอร์ก

– mtNoObjects บ่งจำนวนของออปเจ็กต์สูงสุดที่ใช้ในขณะเดียวกัน ออปเจ็กต์คือ ปากกา (pen) แปรง (brush) ฟอนต์ (font) และบิตแม็พ (bitmap)

– mtMaxRecord บอกขนาดของเรคอร์ดที่ใหญ่ที่สุด (คงเพื่อการจองหน่วยความจำ โดยที่สิ่งที่ต้องการทราบมากกว่าคือ จำนวนเรคอร์ดในไฟล์แต่กลับไม่บอก และฟิลด์นี้ก็แบ่งเป็น 2 เวอร์ก ได้อย่างสบาย)

– mtNoParameters เป็นฟิลด์สำรอง

แถวของเรคอร์ด

ถัดจากส่วนหัวไปคือ แถวของเรคอร์ดเรียงลำดับตามการบันทึก แต่ละเรคอร์ดมีแบบข้อมูลเป็น TMetaRecord ในรูปที่ 5 คือ

– rdSige เป็นขนาดของเรคอร์ดเป็นเวอร์ก

– rdFunction เป็นหมายเลขของฟังก์ชัน (Function number) ตามที่ทำเป็นค่าคงตัวไว้ขึ้นต้นด้วย meta__ ตามด้วยชื่อฟังก์ชันเช่น meta__Rectangle

– rdParam เป็นอะเรย์ไม่จำกัดขนาดของเวอร์ก จำนวนช่องของอะเรย์ขึ้นอยู่กับจำนวนของอาร์กิวเมนต์ของฟังก์ชัน และจะเก็บค่าตามการเก็บลงสแต็กคือ จากท้ายเข้ามา เช่น เมื่อใช้คำสั่งเป็น

```
Ellipse (adc, 100, 200, 300, 400);
```

จะได้ค่าในเรคอร์ดนี้เป็น

```
7, 1048, 400, 300, 200, 100
```

ค่า 1048 คือค่าหมายเลขฟังก์ชันของ Ellipse และที่เห็นมีเลขอยู่เพียง 6 จำนวนเพราะ 7 เป็น LongInt

แม็พปิงโหมด (Mapping mode)

แม็พปิงโหมดหรือถ้าคิดว่าเป็นมาตราอาจทำให้เข้าใจง่ายกว่า ซึ่งได้กำหนดมาตราสำหรับการแสดงภาพ (ทั้งจอภาพและเครื่องพิมพ์) ไว้ทั้งหมด 8 มาตราตามรูปที่ 6 มีชื่อเรียกตามชื่อค่าคงตัวทั้งแต่ mm__Teset ถึง mm__AnIsotropic (ค่า 1 ถึง 8) ดังเช่นเมื่อเราใช้คำสั่ง

```
LineTo (adc, 208, 156);
```

ในความคิดของผู้เขียนโปรแกรมคือ ให้ลากเส้นตรงไปยังจุด 208, 156 กับดีไวซ์คอนเท็กซ์ ซึ่งส่วนมากคือหน้าต่าง แต่วินโดวส์จะถือว่าเป็นการลากเส้นในหน้าต่างสมมติก่อน (ไม่มีตัวตนและไม่มีการลากเส้นจริง) แล้วจึงนำมาลากเส้นในหน้าต่างจริง

จึงสุดแต่ว่าเราจะกำหนดใช้มาตราใดไว้เช่น ให้เป็น mm__LoMetric ค่า 208, 156 คือ จำนวนจุดในมาตรานี้ (ที่แสดงอยู่ทางซ้าย) เมื่อเทียบกับเป็นของจอภาพ (คือทางขวา) จะได้เป็น 64, –48 ค่าลบหมายถึง มาตราพวกนี้คิดค่าในแกน y เป็นบวก ในทางขึ้น ส่วนของจอภาพเป็นบวกในทางลง จึงจะไม่เห็นเส้นนี้ หากต้องการเห็นในหน้าต่างจะต้องใช้คำสั่งเป็น

```
LineTo (adc, 208, –156)
```

โดยที่ในการเขียนโปรแกรมให้ผู้ใช้วาดภาพ เมื่อให้ผู้ใช้กำหนดว่าจะใช้มาตราใดแล้ว (หากให้กำหนด) เช่น เมื่อผู้ใช้ต้องการวาดรูปสี่เหลี่ยมผู้ใช้จะคลิ๊ก-เม้าส์ที่มุมด้านหนึ่ง ซึ่งเราจะอ่านค่ามา

Map mode	X axis	Y axis	
mm_Text	1 / 1	1 / 1	= 96 / 1 inch (VGA)
mm_LoMetric	2080 / 640	1560 / -480	= 10 / 1 mm
mm_HiMetric	20800 / 640	15600 / -480	= 100 / 1 mm
mm_LoEnglish	325 / 254	325 / -254	= 100 / 1 inch
mm_HiEnglish	1625 / 127	1625 / -127	= 1000 / 1 inch
mm_Twips	2340 / 127	2340 / -127	= 1440 / 1 inch
mm_Isotropic	wpx / vpx	wpy / vpy	
mm_AnIsotropic	wpx / vpx	wpy / vpy	{ x <> y }

รูปที่ 6 มาตรา (mapping made) ในการแสดงภาพ

เก็บไว้เป็นจุดต้นทาง เลื่อนเม้าส์ไปยังมุมอีกด้านหนึ่งแล้วปล่อยมือ ซึ่งเราจะต้องเก็บค่าไว้เป็นจุดปลายทาง ค่าของจุดทั้งสองนี้เป็นค่าของหน้าต่าง แต่ขอเรียกเป็นของจอภาพ หากนำไปใช้โดยตรงจะเกิดกรณีดังกล่าวมา ในการนี้ในวินโดวส์ได้เตรียมฟังก์ชัน DPtoLP (และ LPtoDP) ไว้ให้เราเปลี่ยนจุดทั้งสองเป็นของมาตราก่อน ซึ่งจะเปลี่ยนทั้งค่าและเครื่องหมาย แล้วจึงนำไปใช้กับคำสั่ง Rectangle

คำว่า DP ย่อมาจาก Device point คือ จำนวนจุดในหน้าต่างจริง ส่วน LP ย่อมาจาก Logical point คือ จำนวนจุดในหน้าต่างสมมติ

มาตราตามกำหนด

มาตราทั้งแปดตามที่กล่าวมาจะเห็นว่ามีอยู่ 2 มาตราที่ไม่ให้ค่าไว้คือ mm__Isotropic (สมมาตร) กับ mj__AnIsotropic (ไม่สมมาตร) เป็นมาตราที่ให้ผู้เขียนโปรแกรมกำหนดค่าสเกลเองโดยเตรียมฟังก์ชันไว้ 2 ฟังก์ชัน คือ

```
SetWindowExt (adc, wpx, wpy)
SetViewportExt(adc, vpx, vpy)
```

ฟังก์ชันทั้งสองนี้จะต้องใช้คู่กันโดยเรียกใช้ SetWindowExt ก่อน คำว่า Window ในที่นี้หมายถึง หน้าต่างสมมติ ส่วน Viewport หมายถึง หน้าต่างจริง ค่า wpy และ wpy คือ จำนวนจุดในหน้าต่างสมมติ ในแกน x และ y คือตรงกับ LP ส่วน vpx และ vpy หมายถึง จำนวนจุดในหน้าต่างจริง ในแกน x และ y คือตรงกับ DP (ที่ว่าตรงกับ LP และ DP หมายถึงชื่อตรงกัน) หากเทียบกับมาตรา mm__LoMetric จะได้ว่า

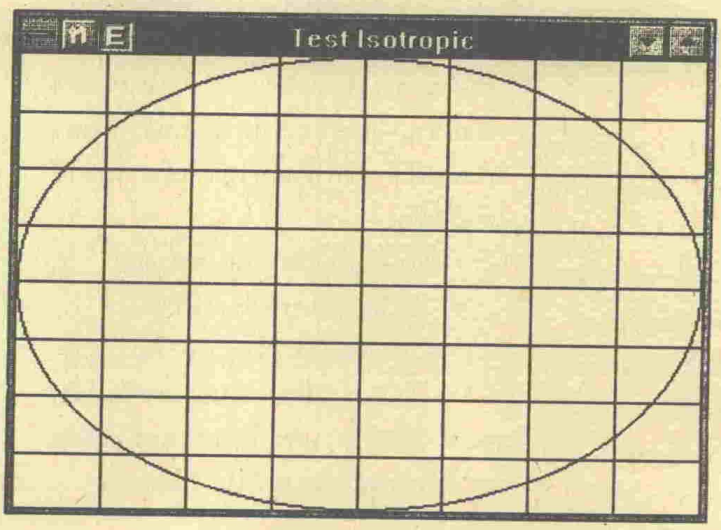
```
wpx = 2080 และ wpy = 1560
vpx = 640 และ vyp = −480
```

```
PROCEDURE  WndProC.Paint(paintdc : HDC;
                var paintInfo : TPaintStruct);
VAR    ix ,iy : Integer;  rct : TRect;
BEGIN
  GetClientRect(HWindow, rct);
  SetMapMode(paintDC, mm_AnisoTropic);
  SetWindowExt(paintDC, 8, 8);
  SetViewportExt(paintDC, rct.right, rct.bottom);

  Ellipse(paintDC, 0,0,8,8);

  for  ix := 1 to 7  do
    begin  MoveTo(paintDC, ix, 0);
           LineTo(paintDC, ix, 8)  end;
  for  iy := 1 to 7  do
    begin  MoveTo(paintDC, 0, iy);
           LineTo(paintDC, 8, iy)  end;
END;
```

ลิสติงที่ 2



รูปที่ 7
ผลการรันของโปรแกรมในลิส

แต่การใช้งานมาตราทั้งสองนี้ส่วนมากจะให้แกน y เป็นบวกในทางลงคือมีค่า vpy เป็นบวก ดูตัวอย่างดีกว่าครับตามโปรแกรมในลิสติงที่ 2 (ซึ่งจะต้องต่อเติมให้เป็นโปรแกรมที่สมบูรณ์) เมื่อรันโปรแกรมนี้จะได้ผลการรันตามรูปที่ 7 ทั้งนี้เพราะว่า

– กำหนดให้ใช้มาตรา mm-AnIsotropic มีผลตามที่จะกล่าวต่อไป

– กำหนดให้หน้าต่างทั้งบานไม่ว่าจะกว้างหรือแคบขนาดไหน ให้มีขนาดของหน้าต่างสมมติเป็น 8×8 LP

– ดังนั้นเมื่อให้วาดรูปวงกลมเต็มหน้าต่าง ผลที่ได้จะเป็นวงรี หากหน้าต่างจริงคือ ตรงพื้นที่ใช้งานไม่เป็นสี่เหลี่ยมด้านเท่า

– ให้ตีตารางขนาด 8×8 คือ 64 ช่อง ดังนั้นไม่ว่าหน้าต่างจริงจะแคบกว้าง ใหญ่เล็กประการใด ก็จะได้ตาราง 64 ช่อง เต็มหน้าต่างจริงเสมอ

ถ้าเปลี่ยนมาตราเป็น mm____Isotropic จะได้วงกลมเป็นวงกลมเสมอ (อาจสัมผัสกรอบหน้าต่างเพียง 2 ด้าน) ตารางจะเป็นตารางสี่เหลี่ยมด้านเท่า (จึงมีที่ว่างในหน้าต่าง) ต้องทดลองดูครับ

ไฟล์ซ้อนไฟล์

ผมคงจะต้องขอยุติเรื่องไว้เพียงเท่านี้ก่อน แต่สำหรับท่านที่ต้องการเขียนโปรแกรมทดลองใช้งานเมตาไฟล์ การบันทึกต่อท้ายไม่ได้ เป็นเรื่องที่ไม่สนุกเลย จึงขอนำโปรแกรมส่วนนี้มาแสดงก่อนตามลิสติงที่ 3 โดยที่ SimMtfC คือ สกุลที่ทำไฟล์ให้เป็นไฟล์ และ RecToPlay คือ โปรแกรมส่วนนี้

เป็นโปรแกรมที่เกี่ยวข้องอยู่กับการอิดิตภาพ เช่น เมื่อให้ผู้ใช้วาดภาพเก็บไว้ในเมตาไฟล์ภาคบันทึก เมื่อต้องเพนต์ใน Paint ของโปรแกรมหน้าต่างที่ใช้สกุลนี้ จะเรียกใช้ Replay ซึ่งจะตรวจว่ามีการบันทึกภาพใหม่หรือไม่ ถ้ามีจะเรียกใช้ RecToPlay โดยผ่านค่าทั้ง recDC และ hPlay มาให้การดำเนินการคือ

– ถ้า hPlay ไม่ใช่ไฟล์เปล่าจะยกเลิกไฟล์เดิม

– ถ่ายทอดภาพจาก recDC ให้ hPlay ด้วยการปิดไฟล์

– ให้กำเนิดไฟล์ภาคบันทึกคือ recDC ใหม่เป็นไฟล์ในหน่วยความจำ (การให้กำเนิดครั้งแรกก็ต้องให้เป็นไฟล์ในหน่วยความจำ)

– ถ่ายทอดภาพกลับมาให้แก่ recDC

เมื่อกลับไป Rcplay จะนำภาพใน hPlay ออกแสดง ด้วยวิธีการเช่นนี้จะทำให้ผู้เขียนโปรแกรมใช้งานสกุล SimMtfC มองสกุลนี้เหมือนไฟล์ธรรมดา คือ เมื่อต้องการบันทึกก็บันทึกไป เมื่อต้องการแสดงก็ให้แสดง เป็นการตัดภาระในการเขียนโปรแกรมต่อ ๆ ไปได้อย่างมาก

เป็นที่น่าประหลาดใจครับทั้งในคู่มือวินโดวส์และในหนังสือต่าง ๆ (เท่าที่ผมอ่าน) ไม่กล่าวถึงเรื่องนี้เลย (การเพลย์เมตาไฟล์ให้แก่เมตาไฟล์ด้วยกัน) คงกล่าวดังในลิสติงที่ 1 คือ บันทึกเพียงครั้งเดียวเก็บลงไฟล์ในดิสก์ นำออกแสดงใหม่แล้วเลิกกัน ทำให้ผมสงสัยว่าวิธีการที่ผมนำมาใช้ถูกต้องหรือไม่ ถ้ามีวิธีอื่นก็น่าจะบอกกันบ้าง หรือทำไม่ได้ (โปรแกรมของผมใช้ได้) ซึ่งถ้าบันทึกต่อไม่ได้เมตาไฟล์ก็ไม่ใช่ไฟล์แห่งอนาคตครับ

ท้ายตอน

สำหรับฉบับนี้คงต้องจบกันเพียงเท่านี้ เพราะโปรแกรมของ SimMtfC ยาวพอสมควร หากรวมกับตอนนี้ก็จะยาวไป พบกันตอนต่อไปทำไฟล์ให้เป็นไฟล์ เพื่อให้เป็นไฟล์ของอนาคตสมกับคำว่า meta ครับ

ลิสติงที่ 3

```
PROCEDURE  SimMtfC.Replay(adc : HDC);
BEGIN
 { if  pctsi[px1] <> pctsi[px2]  then }
  if  vporg.x <> vpxy2.x  then
    Rectangle(adc, vporg.x,vporg.y, vpxy2.x,vpxy2.y);
  WriteFirstRcrd(adc, false);
  if  hvenewpat  then
    begin  RecToPlay(recDC, hPlay);
           sveok := false;
           hvenewpat := false;  hveplay := true  end;
  if  hveplay  then  PlayMetaFile(adc, hPlay);
END;


PROCEDURE  SimMtfC.RecToPlay(var srcRecDC : HDC; var dstPlayH : THandle);
BEGIN
  if  dstPlayH <> 0  then  DeleteMetaFile(dstPlayH);
  dstPlayH := CloseMetaFile(srcRecDC);
  srcRecDC := CreateMetaFile(nil);
  PlayMetaFile(srcRecDC, dstPlayH)
END;
```